LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-644263

Monitoring Of The National Ignition Facility Integrated Computer Control System*

J. Fisher, M. Arrowsmith, E. Stout

September 26, 2013

ICALEPCS 2013
San Francisco, CA, United States
October 6, 2013 through October 11, 2013

Disclaimer

MONITORING OF THE NATIONAL IGNITION FACILITY INTEGRATED COMPUTER CONTROL SYSTEM*

J. Fisher, M. Arrowsmith, E. Stout, LLNL, Livermore, CA, 94550, U.S.A.

Abstract

The Integrated Computer Control System (ICCS), used by the National Ignition Facility (NIF) provides comprehensive status and control capabilities for operating approximately 60,000 devices through 2,600 processes located on 1,800 servers, front end processors and embedded controllers.[1] Understanding the behaviors of complex, large scale, operational control software, and improving system reliability and availability, is a critical maintenance activity. In this paper we describe the ICCS message logging framework, with tunable detail levels and automatic rollovers, and its use in analyzing system behavior. ICCS recently added Splunk as a tool for improved archiving and analysis of these log files (about 50GB, or 35 million logs, per day). Splunk now continuously captures all ICCS log files for both real-time examination and exploration of trends. Its powerful search query language and user interface provides interactive exploration of log data to visualize specific indicators of system performance, assists in problems analysis, and provides instantaneous notification of specific system behaviors.

THE MSGLOG FRAMEWORK

A core framework of ICCS is the MsgLog ("message log") API for recording status messages to a file system. Implemented in both Ada and Java, all ICCS software layers use this logging engine.

Log files are designed to be rolling, with old messages on the file system removed at regular intervals. Each ICCS process, through configuration data, specifies the number of log files to rotate through, and the maximum size of each log file. To allow viewing of start-up behavior, the first log file generated is not part of the rotation process. A default behavior of four log file, 10,000 lines each, is used if no overriding configuration data is provided. Here is a sample log entry:

```
  2013/06/17 15:14:21.8973 T_LOG
OPG|Q25T|AMC|Shape_Sys (TID 0055) Entering
Setpoint_Monitors.Create_Mappers to map taxon
OPG|Q25T|AMC|Shape_Sys
```

The first two fields are the date and time stamp. The T_LOG identifier indicates this is a Tertiary log (typically just for fine-grain debugging). Other possible values include P_LOG (Primary), S_LOG (Secondary), and E_LOG (Exceptions, or Errors). Exception logs are only used for significant failures. Primary logs document key system changes that can be used for incident analysis (e.g. motor moves). Secondary and Tertiary logs are non-critical, and left to the discretion of the system manager and software developer. By default, tertiary logs are disabled in production, and only enabled for troubleshooting.

The next field, "OPG|Q25T|AMC|Shape_Sys", referred to as a taxon, specifies the entity that made the log entry. Taxons are used as the ICCS naming system and each represents a unique control point in NIF. The next field, "(TID 0055)" specifies the thread that is writing this log entry. The content that follows the thread is a description of what happened.

In terms of coding, an Ada log entry would look like this (date/time and thread are determined automatically):

```
  Msg_Log_Api.Log_Exception("Unexpected
exception releasing AMC LCU", Get_Taxon (Self));
```

For the Java code base, a log entry would look like this:

```
  log.logException("Unable to initialize
controller", ex);
```

The taxon value is automatically populated by the Java Framework. The parameter ex is a Java exception object; the corresponding stack trace will automatically be logged as well.

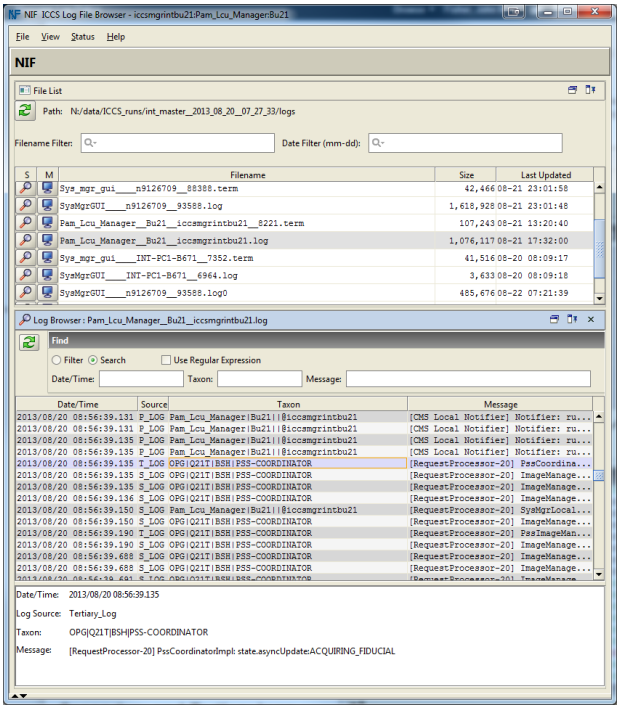

Figure 1: The MsgLog Browser provides graphical navigation of ICCS logs at runtime

*This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. #LLNL-ABS-632634

LOG FILE ANALYSIS BEFORE SPLUNK

For online viewing of log files in the NIF, operators and developers can either look at the raw logs stored on the file system, or navigate to the logs through a graphical Msg Log Browser (Figure 1). This browser provides basic parsing and filtering tools to better analyze system behavior.

In the 24x7 NIF environment, issues come up periodically that require a deeper offline analysis of the log data produced by ICCS. Because the logs are rolling, ICCS developed a "snapshot" capability, to archive the logs as they existed when (or shortly after) the incident of concern occurred. Typically ICCS will produce about 8,500 log files (about 18 GB of data) accumulated when the system has been up for a few days (i.e. all process logs are rolling over). Fortunately, the 3Par file system provides a low-level snapshot capability, where the entire log directory can be duplicated extremely quickly. All the content of the copied log directory is then compressed into a ZIP file, and transferred to the ICCS development network file system for review. Developers then either browse the ZIP file using the Msg Log Browser, or extract the files as needed and browse them with text editors.

While this snapshot mechanism served ICCS well for quite a number of years, it has some distinct limitations. Most significantly, the rolling behavior of the logs meant that many logs were simply lost after days or even hours. When problems occurred (particularly during off hours), a quick response was needed by NIF operators. But, several manual steps were required for developers to access the log data. The analysis tools were primitive – searching manually through individual text files, or post-processing the file with Perl scripts, and using Excel for further analysis. Because of the huge quantities of log file data, and the difficulty in exploring the data, some problems simply remained overlooked.

SPLUNK

Fortunately, the NIF IT organization had begun exploring a tool called Splunk for analyzing log files generated by the thousands of computers that comprise the NIF computing infrastructure. Splunk, developed by the company of the same name, provides comprehensive tools for archiving, indexing/searching, parsing, analyzing, and visualizing tremendously large quantities of log file data.

Splunk was rapidly recognized as a well suited engine for better understanding ICCS log data. Today, all log data generated by NIF ICCS, as well as the ICCS QA and Integration environments, is stored in Splunk – about 50GB per day in total. Splunk is currently configured to retain log data from the last 45 days.

Configuring Splunk for ICCS

Splunk captures logs by actively monitoring a parent directory – all output to log files, and the creation of new files, are immediately recognized, and the log data is pulled into a data repository outside the control system network. Now, the rollover of ICCS log files is irrelevant – logs are captured as they occur and archived automatically.

Each log entry has a variety of log data built in to the filename and path of the file. Consider the filename:

```
.../ICCS_runs/int_master__2013_02_04__07_56_55/l
ogs/emulated_pam_fep__Q16B__iccsint001.log0
```

Splunk is configured to automatically recognize that the ICCS instance is "int_master", the program is "emulated_pam_fep", the location is "Q16B", and the host generating the log is "iccsint001". For each log entry, Splunk will also automatically recognize the date/time of occurrence, the taxon, and the thread, based on regular

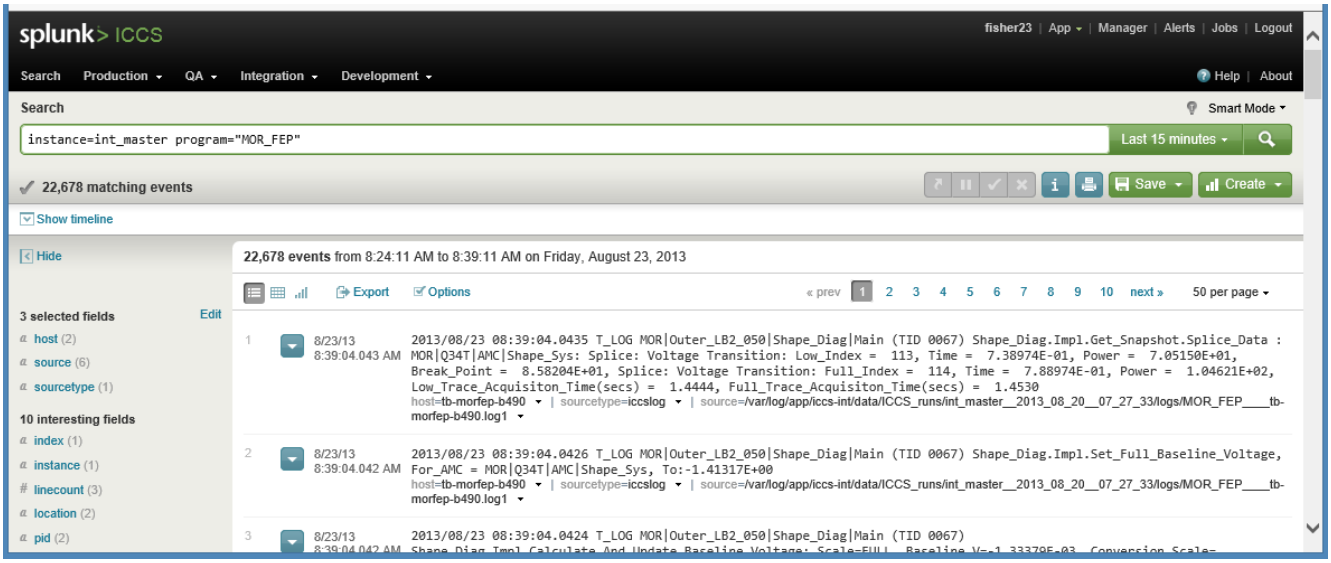

Figure 2: The Splunk user interface, with a view of recent log files generated by the MOR_FEP program

expressions in configuration files

Splunk provides a search language to allow arbitrarily complex data mining. Figure 1 depicts a simple search query into the ICCS log data. Displayed are all log entries generated in the last 15 minutes by the MOR_FEP running in the Integration (int_master) instance of ICCS. Typically, the lag between log generation and viewing in Splunk is less than five seconds.

ANALYSIS EXAMPLES

The subsections below are a small sampling of the ways that ICCS developers and users leverage Splunk to better understand control system behaviors.

Characterizing Error Logs

With 2,600 programs running concurrently, it has been historically difficult to characterize the broad behaviors of ICCS. All these programs generate diagnostic logs, but which programs are generating a significant number of errors? How often? Splunk now makes such questions trivial to answer. Consider the following simple query:

```
instance="nif" E_LOG | top program
```

The query is a data cascade of two search commands (separated by "|"). The first command (which is always an implied search) will output all log entries containing the text E_LOG, in the instance of ICCS called "nif". The 'top' command then summarizes the output, providing the most common values for the given field (program). The results, over a 15 minute period, are shown in Figure 3. The Splunk user interface provides a drilldown capability; when the user clicks on any blue text of the first row, for example, the query will automatically become:

```
instance="nif" E_LOG program="MOR_FEP"
```

The user will then be presented a list of actual log entries, similar to Figure 2.

	program ≑	count ≑	percent ≑
1	MOR_FEP	195	41.666667
2	tapfep	88	18.803419
3	acfep	86	18.376068
4	Config_Services	57	12.179487
5	arcfep	10	2.136752
6	PEPC_FEP	9	1.923077
7	Navigator	8	1.709402
8	Power_Cond_FEP	6	1.282051
9	Segment_Manager	4	0.854701
10	Target2_Diag_FEP	3	0.641026

Figure 3: A count of exception logs, by program, over a 15 minute period

Automatic Alignment Loop Times

For each NIF experiment, ICCS executes a complex series of steps to correctly align all 192 laser beams. A step, or "loop" takes a certain amount of time to perform. NIF operators have observed that the loop times were increasing over a series of days, to the point where timeout failures were occurring. When a loop executes, a log is generated. For example:

```
2013/08/17 02:15:06.959 S_LOG
Segment_Manager|Bu45||@iccsmgrprodbu45
[Interpreter AA|B453|FOA|SEG-MGR] Metric:
[Interpreter AA|B453|FOA|SEG-MGR] Metric: @@@
SEGMGR AA_BEAM_TO_TAS_COARSE LOOP_EXECUTION
[4.4681] AA|B453|FOA|SEG-MGR
```

In the log, "AA_BEAM_TO_TAS_COARSE" is a loop name, and the number in brackets is the time in seconds it took the loop to complete. Here is a Splunk query to visualize loop performance:

```
instance=nif program=segment_manager
"AA_BEAM_TO_TAS_COARSE LOOP_EXECUTION" |
rex "LOOP_EXECUTION \[(?<duration>.+)\]" |
timechart span=4h max(duration)
```

The first command narrows the logs to those generated by the segment_manager programs that contain the text "AA_BEAM_TO_TAS_COARSE LOOP_EXECUTION". The second command does a Perl-like regular expression on the previous results, and extracts the loop execution time into a new field called "duration". The third command generates a graphical chart of "duration" values over a span of several days (the date/time range is adjustable through the user interface). The maximum duration seen during each four hour period is plotted. The chart in Figure 3 demonstrates the query results.

The problem was localized to performance degradation on VxWorks Front End Processors (FEPs) used for motor movement. A periodic restart of specific FEPs minimized this performance degradation (longer term, VxWorks is being replaced with Linux). The last bar in Figure 3 demonstrates the results after restarting several specific FEPs.

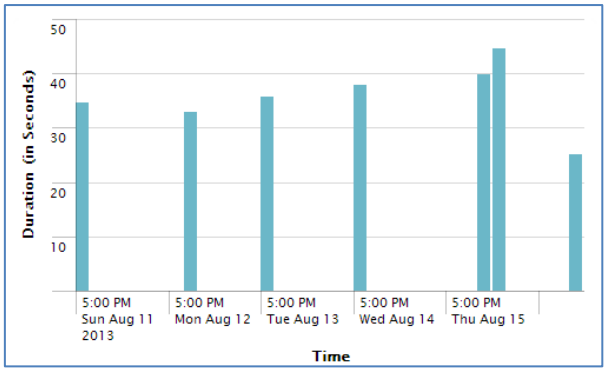

Figure 3: Performance of an Automatic Alignment loop, over several days

Experiment Archiving

The log entry below is generated each time a device in ICCS archives data to a database repository, following the execution of an experiment (or "shot"):

```
2013/08/06 23:59:57.701 S_LOG
Archive_Server|||@iccsprod0002 [WorkerThread 26
(Database Transaction Mgr)] Metric:
[WorkerThread 26 (Database Transaction Mgr)]
Metric: @@@ request_Data_Archive Worker [0.0297]
shotId=C130805-AA
diagId=AggregatePeriodicArchiver
requestId=137585879767000.000
formatter=nif.subsystems.cts.db.SIM960AggregateF
ormatter done
```

Depending on the data being archived, archiving can take a very long time. A Splunk query allows ICCS to characterize which formatters (code used to archive data) are the slowest, on average:

```
instance=nif program=Archive_Server "@@@
request_Data_Archive" |
rex "Worker \[(?<duration>.*)\]" |
search shotId=N130804-001-999 |
stats avg(duration) by formatter
```

Only a particular experiment (N130804-001-999) is looked at. The results are shown as a table in Figure 5. This information is valuable in improving the overall post-shot data archiving time.

	formatter	avg(duration)
1	nif.subsystems.pcs.db.CurrentMonFormatter	47.554560
2	nif.common.energy_diag.db.CalDataFormatter	0.300675
3	nif.subsystems.pcs.db.ChargeProfilesFormatter	0.148952
4	nif.subsystems.pcs.db.ShotDataFormatter	0.107205
5	nif.subsystems.pcs.db.ShotSetupFormatter	0.081474

Figure 5: The devices that take the most time to archive data following a NIF experiment

Server Monitoring

All ICCS Framework and Supervisory servers are monitored by Oracle Enterprise Manager (OEM) [2]. System performance metrics are stored in a relational database once every five minutes.

Splunk provides the ability to mine and analyze database tables just like log data. For example, the Splunk query below gathers OEM metrics data, and plots the results in a timeline:

```
| dbquery OEM "SELECT
regexp_replace(d.target_name, '\\..*', '') AS
Host, d.collection_timestamp, d.metric_column as
Metric_Type, TO_NUMBER(d.value) as Metric FROM
sysman.mgmt$metric_details  d WHERE
d.collection_timestamp >= SYSDATE -
$selectedDays$ and d.target_type = 'host' and
d.target_name like 'iccsfwprod%' and
d.metric_column = '$selectedReport$'" |
rename COLLECTION_TIMESTAMP to _time |
timechart span=$increment$ max(METRIC) by HOST
```

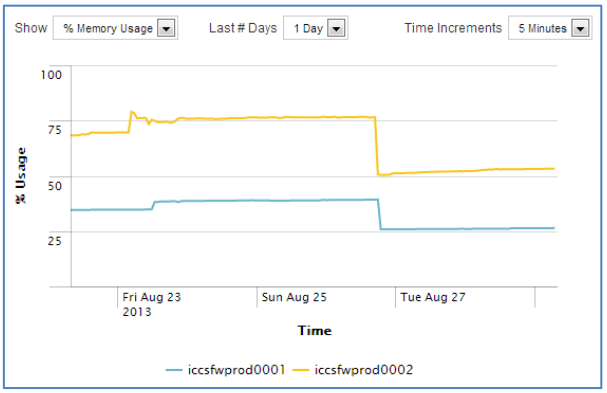

Figure 6: Leveraging Oracle Enterprise Manager, Splunk is used to analyze system performance metrics stored in database tables

This particular query resides in an XML file, which is a more advanced capability in Splunk for creating application-like dashboards. The query is parameterized ($selectedDays$, $selectedReports$, $increment$), based on user fields on the dashboards. The user fields and the resulting chart are shown in Figure 6.

The two hosts in this graph are both Oracle Virtual Machines (OVMs); as a result of this analysis, memory for both were increased (as can be seen in the graph). OVM technology allowed this to be done without a server restart.

CONCLUSION

A comprehensive message logging engine was developed for ICCS at its initial development. The log files are instrumental to understanding complex system behaviors, but mining the log files was slow and difficult until the introduction of Splunk.

Splunk now greatly enhances ICCS performance analysis capabilities. Its powerful user interface allows for ad-hoc browsing as well as application-like dashboards. It is now much easier to evaluate and enhance ICCS performance, resulting in improved usability and performance of the NIF.

REFERENCES

[1] P. Van Arsdall, et al, "National Ignition Facility Project Completion and Control System Status," ICALEPCS'09, Kobe, Japan, Oct 2009, TUP078, p. 260 (2009); http://www.JACoW.org.

[2] T. Frazier, et al, "Optimizing and Automating Virtual Infrastructure for Fusion Energy Research," Oracle OpenWorld 2012, October 2012, SBH7169; http://www.oracle.com/openworld.